Verification and (some) Cryptography

Wednesday 9:30-11:45

Aurojit Panda

What is Verification?

Prove **properties** about **programs.**

What is Verification?

Prove **properties** about **programs.**

Example: Keynote does not crash given inputs from remote.

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)
  - If assumptions are minimal: **stronger** guarantee than testing.

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)
  - If assumptions are minimal: **stronger** guarantee than testing.
  - Useful for critical systems like **networks**

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)
  - If assumptions are minimal: **stronger** guarantee than testing.
  - Useful for critical systems like **networks**
- **Original** question for computer science

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)
  - If assumptions are minimal: **stronger** guarantee than testing.
  - Useful for critical systems like **networks**
- **Original** question for computer science
  - Turing: Does a given Turing machine halt?

Why Care about Verification?

- **Guarantee correctness** of programs (modulo some assumptions)
  - If assumptions are minimal: **stronger** guarantee than testing.
  - Useful for critical systems like **networks**
- **Original** question for computer science
  - Turing: Does a given Turing machine halt?
  - Church: Are two statements in lambda calculus equivalent?

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.
  - **Example**: Approximate a **program** as a **finite state machine**.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.
  - **Example**: Approximate a **program** as a **finite state machine**.
- Even when possible **search** over **large space**, e.g. sequences of packets.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.
  - **Example**: Approximate a **program** as a **finite state machine**.
- Even when possible **search** over **large space**, e.g. sequences of packets.
  - **Approach**: **Domain knowledge** to simplify search.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.
  - **Example**: Approximate a **program** as a **finite state machine**.
- Even when possible **search** over **large space**, e.g. sequences of packets.
  - **Approach**: **Domain knowledge** to simplify search.
  - **Example**: Show that packet order does not matter.

Why Hard?

- Some problems are proven to be **impossible,** e.g., halting problem.
  - **Approach**: **Approximate** to make verification possible.
  - **Example**: Approximate a **program** as a Assumptions about system.
- Even when possible **search** over **large space**, e.g. sequences of packets.
  - **Approach**: **Domain knowledge** to simplify search.
  - **Example**: Show that packet order does not matter.

Network Verification!

Verification Papers Questions

- **What assumptions** are required?

Verification Papers Questions

- **What assumptions** are required?
  - False negatives (soundness) & false positives (completeness).

Verification Papers Questions

- **What assumptions** are required?
  - False negatives (soundness) & false positives (completeness).
- **How** is the problem **encoded**?

Verification Papers Questions

- **What assumptions** are required?
  - False negatives (soundness) & false positives (completeness).
- **How** is the problem **encoded**?
  - Verification complexity, & tools?

Verification Papers Questions

- **What assumptions** are required?
  - False negatives (soundness) & false positives (completeness).
- **How** is the problem **encoded**?
  - Verification complexity, & tools?
- How do they **scale**?

Verification Papers Questions

- **What assumptions** are required?
  - False negatives (soundness) & false positives (completeness).
- **How** is the problem **encoded**?
  - Verification complexity, & tools?
- How do they **scale**?
- How much **manual effort** is needed?

A Formally Verified NAT

Arseniy Zaostrovnykh
EPFL, Switzerland
arseniy.zaostrovnykh@epfl.ch

Solal Pirelli
EPFL, Switzerland
solal.pirelli@epfl.ch

Luis Pedrosa
EPFL, Switzerland
luis.pedrosa@epfl.ch

Katerina Argyraki
EPFL, Switzerland
katerina.argyraki@epfl.ch

George Candea
EPFL, Switzerland
george.candea@epfl.ch

Programs

- **Network functions**
  - NAT

Properties

- Correctly implements RFC 3022
- Does not crash.
- Does not leak memory, etc.

What is New?

- Work on **network function verification** normally relies on **models**.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.
- **Assumption**: NAT complexity largely lies in data structure not forwarding.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.
- **Assumption**: NAT complexity largely lies in data structure not forwarding.
- **Key idea**: separately verify correctness for **data structures** and **forwarding**.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.
- **Assumption**: NAT complexity largely lies in data structure not forwarding.
- **Key idea**: separately verify correctness for **data structures** and **forwarding**.
  - Data structures: **hand written, mechanically checked proofs**.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.
- **Assumption**: NAT complexity largely lies in data structure not forwarding.
- **Key idea**: separately verify correctness for **data structures** and **forwarding**.
  - Data structures: **hand written, mechanically checked proofs**.
  - Forwarding: **symbolic execution**.

What is New?

- Work on **network function verification** normally relies on **models**.
  - Assumes human can accurately translate **model** to **code.**
- **This work**: Directly verify implementation of a NAT.
- **Assumption**: NAT complexity largely lies in data structure not forwarding.
- **Key idea**: separately verify correctness for **data structures** and **forwarding**.
  - Data structures: **hand written, mechanically checked proofs**.
  - Forwarding: **symbolic execution**.
- **Main Result**: How to combine these two types of proofs.

A General Approach to Network Configuration Verification

Ryan Beckett
Princeton University

Aarti Gupta
Princeton University

Ratul Mahajan
Microsoft Research & Intentionet

David Walker
Princeton University

Programs

- **Control plane configuration**
  - BGP/OSPF/routing protocols

Properties

- Computed paths have no loops.
- Reachability/Isolation.
- Traffic not blackholed.
- Two paths are equal length.

What is New?

- **Different encoding** of the problem compared to existing tools.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.
  - Vertex represent **routing protocol** at a **router**.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.
  - Vertex represent **routing protocol** at a **router**.
  - Edge represent that two protocols **might exchange** messages.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.
  - Vertex represent **routing protocol** at a **router**.
  - Edge represent that two protocols **might exchange** messages.
  - Use **SMT solver** to find one set of routing messages that lead to violation.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.
  - Vertex represent **routing protocol** at a **router**.
  - Edge represent that two protocols **might exchange** messages.
  - Use **SMT solver** to find one set of routing messages that lead to violation.
- **Assumption**: Understand control plane semantics and how config is used.

What is New?

- **Different encoding** of the problem compared to existing tools.
- Encode control plane as graph.
  - Vertex represent **routing protocol** at a **router**.
  - Edge represent that two protocols **might exchange** messages.
  - Use **SMT solver** to find one set of routing messages that lead to violation.
- **Assumption**: Understand control plane semantics and how config is used.
  - Some additional overapproximation mentioned in the paper.

Pretzel: Email encryption and provider-supplied functions are compatible

Trinabh Gupta[*†] Henrique Fingler[*] Lorenzo Alvisi[*‡] Michael Walfish[†]

[*]UT Austin [†]NYU [‡]Cornell

Cryptography

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?
- **Common answer**: Yes, but no spam filtering, message classification, etc.

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?
- **Common answer**: Yes, but no spam filtering, message classification, etc.
  - Bad for users: Do not get features like spam filtering.

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?
- **Common answer**: Yes, but no spam filtering, message classification, etc.
  - Bad for users: Do not get features like spam filtering.
  - Bad for providers: Can’t recoup costs through advertising.

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?
- **Common answer**: Yes, but no spam filtering, message classification, etc.
  - Bad for users: Do not get features like spam filtering.
  - Bad for providers: Can’t recoup costs through advertising.
- **This Paper**: End-to-End encryption is not anathema to these features.

Quick Overview

- **Problem**: Can we implement end-to-end encryption for email?
- **Common answer**: Yes, but no spam filtering, message classification, etc.
  - Bad for users: Do not get features like spam filtering.
  - Bad for providers: Can’t recoup costs through advertising.
- **This Paper**: End-to-End encryption is not anathema to these features.
  - Use **two party computation** to implement classification with confidentiality.

Conclusion

- Go to the 9:30am session on Wednesday

Conclusion

- Go to the 9:30am session on Wednesday
  - Pros: Interesting area + should already be there for Jen’s talk at 8:30am.

Conclusion

- Go to the 9:30am session on Wednesday
  - Pros: Interesting area + should already be there for Jen’s talk at 8:30am.
- For all these papers (even cryptography) useful to reason about

Conclusion

- Go to the 9:30am session on Wednesday
  - Pros: Interesting area + should already be there for Jen’s talk at 8:30am.
- For all these papers (even cryptography) useful to reason about
  - Assumptions for correctness.

Conclusion

- Go to the 9:30am session on Wednesday
  - Pros: Interesting area + should already be there for Jen’s talk at 8:30am.
- For all these papers (even cryptography) useful to reason about
  - Assumptions for correctness.
  - How solution scales.

Conclusion

- Go to the 9:30am session on Wednesday
  - Pros: Interesting area + should already be there for Jen’s talk at 8:30am.
- For all these papers (even cryptography) useful to reason about
  - Assumptions for correctness.
  - How solution scales.
  - What is missed by the solution: what can it not detect, or what is leaked.